Not All Sequence Tags Are Created Equal: Designing and Validating Sequence Identification Tags Robust to Indels

Brant C. Faircloth[1]*, **Travis C. Glenn**[2]

1 Department of Ecology and Evolutionary Biology, University of California Los Angeles, Los Angeles, California, United States of America, **2** Department of Environmental Health Science, University of Georgia, Athens, Georgia, United States of America

Abstract

Ligating adapters with unique synthetic oligonucleotide sequences (sequence tags) onto individual DNA samples before massively parallel sequencing is a popular and efficient way to obtain sequence data from many individual samples. Tag sequences should be numerous and sufficiently different to ensure sequencing, replication, and oligonucleotide synthesis errors do not cause tags to be unrecoverable or confused. However, many design approaches only protect against substitution errors during sequencing and extant tag sets contain too few tag sequences. We developed an open-source software package to validate sequence tags for conformance to two distance metrics and design sequence tags robust to indel and substitution errors. We use this software package to evaluate several commercial and non-commercial sequence tag sets, design several large sets ($\max_{count} = 7{,}198$) of edit metric sequence tags having different lengths and degrees of error correction, and integrate a subset of these edit metric tags to polymerase chain reaction (PCR) primers and sequencing adapters. We validate a subset of these edit metric tagged PCR primers and sequencing adapters by sequencing on several platforms and subsequent comparison to commercially available alternatives. We find that several commonly used sets of sequence tags or design methodologies used to produce sequence tags do not meet the minimum expectations of their underlying distance metric, and we find that PCR primers and sequencing adapters incorporating edit metric sequence tags designed by our software package perform as well as their commercial counterparts. We suggest that researchers evaluate sequence tags prior to use or evaluate tags that they have been using. The sequence tag sets we design improve on extant sets because they are large, valid across the set, and robust to the suite of substitution, insertion, and deletion errors affecting massively parallel sequencing workflows on all currently used platforms.

Citation: Faircloth BC, Glenn TC (2012) Not All Sequence Tags Are Created Equal: Designing and Validating Sequence Identification Tags Robust to Indels. PLoS ONE 7(8): e42543. doi:10.1371/journal.pone.0042543

Editor: Shin-Han Shiu, Michigan State University, United States of America

Received May 14, 2012; **Accepted** July 9, 2012; **Published** August 10, 2012

Funding: This work was supported by a Smithsonian Scholarly Studies Grant to Stephen P. Hubbell and BCF, National Science Foundation (NSF) grant DEB-1136626 to BCF and TCG, NSF grant DEB-0614208 to TCG, an Amazon Web Services Educational grant to BCF and TCG, and material (TruSeq-style adapters) and sequencing contributions (HiSeq lanes) from Integrated DNA Technologies. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: This study was partly supported by an Amazon Web Services Educational grant. Indexed-adapter (TruSeq-style) and sequencing contributions (HiSeq lanes) from Integrated DNA Technologies supported this work. TruSeq-style Oligonucleotide sequences © 2007–2012 Illumina, Inc. All rights reserved. Derivative works created by Illumina customers are authorized for use with Illumina instruments and products only. All other uses are strictly prohibited. There are no further patents, products in development or marketed products to declare. This does not alter the authors' adherence to all the PLoS ONE policies on sharing data and materials, as detailed online in the guide for authors.

* E-mail: brant@faircloth-lab.org

Introduction

Synthetic, oligonucleotide sequence identification tags (sequence tags) can be attached to individual pieces of DNA allowing pooling and sample tracking during massively parallel sequencing (MPS) [1–3]. Sequence tags enable efficient distribution of the output from these platforms among many individually identifiable samples rather than extensive, deep sequencing of single individuals or mixed samples. Thus, the ability to tag and track sequenced DNA from many individuals in multiplex increases the efficiency of MPS when the genomes being sequenced are small [4] or when researchers want to apportion the output of MPS platforms among smaller genomic regions of many individuals [5–7].

Groundbreaking prior work introduced the idea of sequence tagging by incorporating tags to sequence reads using polymerase chain reaction (PCR) primers and DNA ligation [1–3]. Yet, early sequence tags were designed for specific platforms and platform-specific error patterns, and few tag sets were created to address the complement of errors (insertions, deletions, and substitutions) affecting the uniqueness of each tag sequence across the suite of current sequencing platforms. Errors can also be introduced to sequence tags during tag synthesis and strand replication (library preparation or template amplification), in addition to DNA sequencing.

Errors in sequence tag synthesis occur during the coupling reaction, when DNA bases are being joined to form the desired oligonucleotide strand [8]. Coupling errors produce n-1, n-2, and n-3 congeners containing deletion errors throughout the oligo [9,10]. Relatively expensive purification techniques remove most of these congeners, particularly the n-2 and n-3 varieties, but some n-1 congeners remain, even with increasingly sophisticated purification methods (e.g., HPLC) [11]. Thus, all synthetic oligonucleotides have the potential to contain deletion errors, and this potential increases significantly when expensive purification is not used. However, expensive purification techniques are increasingly cost prohibitive as the number of required sequence tags or adapters containing tags increases, and HPLC purification can introduce additional problems if sequence tagged adapters or sequence tagged primers are sequentially purified [12] without accounting for carryover.

Errors in strand replication often occur during the amplicon generation or library preparation process (*c.f.* [13]), because researchers use thermostable DNA polymerases and PCR to generate amplicons, increase library concentration by ligation-mediated PCR, or add sequence tags to adapter-ligated fragments. Thermostable DNA polymerases predominately incorporate substitution errors to DNA strands during replication [14,15], although most DNA polymerases can produce new DNA strands containing insertion or deletion errors at a lower frequency [15,16]. The error rate is template- and polymerase-dependent, and modern proof-reading DNA polymerases having exonuclease activity exhibit low rates of nucleotide incorporation error, suggesting that these types of enzymes should be used in all amplicon sequencing and library preparation procedures [17]. Similar synthesis errors accrue during downstream template amplification (i.e., emulsion PCR [emPCR] for 454, Ion Torrent and SOLiD platforms or cluster formation for Illumina), but this is generally less of a problem because sequences are determined from the consensus of many molecules on one particle or in one cluster.

Sequencing errors occur on all MPS platforms, but the type of errors and the error rates vary across MPS platforms [18–25]. Sequencing errors on platforms from Roche 454, Applied Biosystems (Ion Torrent), and Pacific Biosciences largely consist of insertion and deletion errors, whereas sequencing errors on platforms from Illumina and Applied Biosystems (SOLiD) are generally substitutions [26,27]. Single-read sequencing error rates vary from 0.5–5% [20,21,25,28] on Roche, Illumina, and Applied Biosystems platforms to 18% on the Pacific Biosciences platform [23]. Sequencing error rates are not uniformly distributed across sequence reads from platforms that amplify the templates (e.g., Illumina, Ion Torrent and Roche) with most errors occurring at the beginning and end of reads [18,22,29]. This biased distribution of sequencing errors along a read affects sequence tags immediately adjacent to or far from the start of the sequence read [30] to a greater degree than sequence tags offset from 5′ or 3′ ends.

Synthesis, replication, and sequencing errors negatively impact the utility of sequence tags because they change the basepair composition of individual tags by inserting bases to, substituting bases within, or deleting bases from the identifying sequence. All three types of error can cause one tag to appear identical to another (crossover) or sufficiently alter a sequence tag such that it is unrecognizable (loss) and untraceable to the source material. A uniformly distributed error rate of 1.0% during an MPS sequencing run producing 10^6 reads, each having an 8 bp sequence tag, results in approximately 77,000 reads (8%) having more than one error within the sequence tag (Figure S1). Probability ensures that longer sequence tags, which allow multiplexing of more samples, are affected by sequencing error to a greater degree, and tags of longer length should have greater minimum distance from all tags in the set.

Using error-correction schemes, researchers can construct sequence tags that are more robust to synthesis, replication, and sequencing errors (i.e., minimizing crossover and loss) while also allowing the correction of certain types of errors. Hamady et al. [31] used Hamming codes [32] to develop a set of error-correcting sequence tags with which they successfully tracked a large number of reads in multiplex (see also [33]). However, Hamming codes assume that the errors occurring within each sequence tag are only substitutions [34,35]. Insertion and deletion errors violate the codeword scheme and reduce the utility of Hamming-based tags when commercial synthesis does not completely remove n-1 congeners, standard *Taq* polymerase is used during strand replication, or sequence data are generated on platforms incorporating insertion and deletion errors (Figure 1; [36]). Additionally, when Hamming-distance tags are constructed using a binary representation of each base (e.g., T = 00; G = 01; C = 10; A = 11), which we define as "binary encoding" (Figure S2), 33% of substitution errors, while detectable, are uncorrectable because sequencing errors occur among actual nucleotides (Figure 2; [37]). Thus, sequence tags appropriately designed using Hamming codes should use nucleotide representations of each base rather than their binary encoding [37].

Sequence tags based on the edit metric or Levenshtein distance [38,39] are superior to Hamming-distance tags, because edit metric sequence tags are robust to the types of errors introduced by oligonucleotide synthesis, replication, and DNA sequencing: insertions, deletions, and substitutions. Edit metric sequence tags allow for error correction according to the following formulas [38–40]:

$$\text{Required Edit Distance} = 2 \times (\text{Errors}) + 1$$

or

$$\text{Correctable Errors} = (\text{Edit Distance} - 1)/2$$

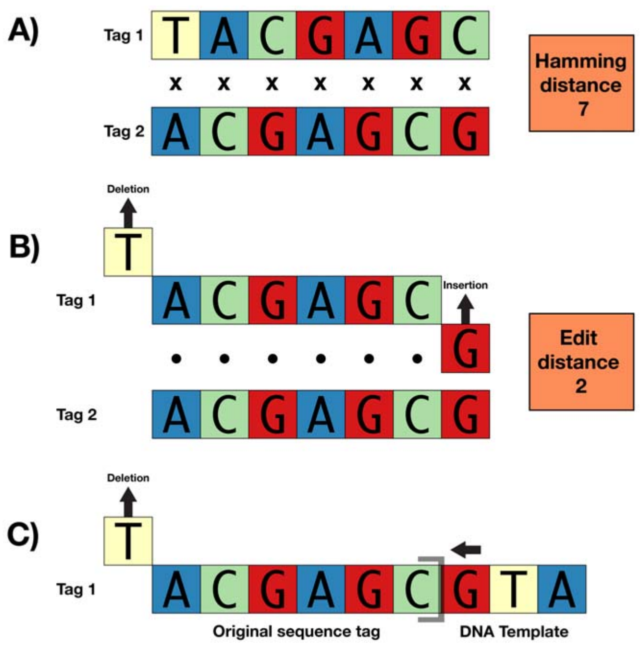

Figure 1. Insertion and deletion errors violate the codeword scheme and reduce the utility of Hamming-based tags. Panel (A) shows two sequence tags that are different from one another by seven substitutions (Hamming distance = 7) – a distance more than sufficient to differentiate tags in the presence of substitution errors. However, these same two tags have an edit distance of two (B) – meaning that a total of two insertions, substitutions, or deletions can turn Tag 1 into Tag 2 and confuse samples. Although it seems improbable that two indels or substitutions would occur in a sequence tag, consider the third case (C) in which a single deletion event at the 5′ end of a sequence tag adjoining DNA template beginning with 5′ guanine confuses Tag 1 with Tag 2. Edit metric sequence tags of distance three or greater would mitigate this mistake.
doi:10.1371/journal.pone.0042543.g001

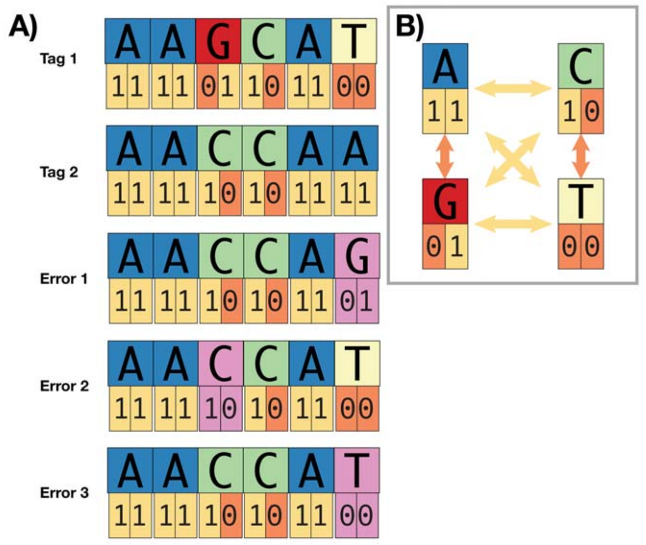

Figure 2. Using Hamming codes to design binary encoded sequence tags when synthesis, replication, or sequencing errors mutate the nucleotide sequence reduces the number of single-base errors that are correctable during downstream demultiplexing. Here, we show two sequence tags (Tag 1 and Tag 2) and both their nucleotide and binary encodings. Tag 1 and Tag 2 have a Hamming distance of four between their binary representations and a Hamming distance of two between their nucleotide representations. Error 1 is correctable to Tag 2, because a single nucleotide substitution (in purple) results in a single, binary difference (11 versus 01) between Error 1 and Tag 2, and single binary errors are correctable when tags are at least three binary differences from each other. Error 2 and Error 3 tags also exhibit a single nucleotide substitution (in purple) but two binary differences from Tag 1 and two binary differences from Tag 2. Because there is more than a single binary difference, we cannot determine whether the source tag was originally Tag 1 or Tag 2, we cannot correct the error, and we must discard the read. More generally, because of the binary encoding and the Hamming distance between tags (Hamming distance four between binary representations, Hamming distance two between nucleotide representations), we can correct single binary errors seen in the substitutions around the perimeter of inset (B), but we cannot correct double binary errors across the diagonals of inset (B). Because these single nucleotide, double binary substitutions (i.e., across the diagonals) comprise two of six potential substitution mutations, we cannot correct 33% (2/6) of single nucleotide substitution errors.
doi:10.1371/journal.pone.0042543.g002

Thus, we can correct up to two sequencing errors in sequence tags from a set having an edit distance of five. Although edit metric sequence tags are provided by several commercial (e.g., Roche 454, Inc.) and non-commercial sources [40,41], there are few available methods (*c.f.* [29]) of generating sets of edit metric-based sequence tags. Furthermore, current methods may generate tags that do not correctly follow the edit metric (Table 1), and current methods are best suited to generating sequence tag sets comprising tags of shorter length (≤8 nt). The continually increasing output of MPS platforms suggests that large collections of edit metric sequence tags will be essential to distributing output across smaller genomes, select genomic regions, and populations of individuals.

Here, we introduce *EDITTAG*, a collection of tools for testing sequence tags for conformance to the edit or Hamming distance metric, generating edit metric sequence tags, and programmatically applying sequence tags to PCR primers and platform-specific sequencing adapters. *EDITTAG* differs from similar programs by providing: (1) a method to check the conformity of previously designed tags, adapters, linkers, or primers to the edit metric; (2) a method to generate edit metric sequence tags of arbitrary length; (3) methods for prepending sequence tags to amplification primers and inserting tags into platform-specific sequencing adapters; and (4) multiprocessing support to speed tag generation when tag lengths are long (≥8 nt).

We use components of EDITTAG to validate a number of existing sequence tag sets provided by commercial and non-commercial sources, design several sets of edit metric sequence tags of varying edit distance, and integrate a subset of edit metric sequence tags to Epicentre Nextera adapters, Illumina TruSeq adapters, and PCR primers. We then validate this subset of tags by sequencing across the indices of indexed adapters and sequence-tagged PCR primers on the Illumina (GAIIx and HiSeq 2000) and Roche 454 (FLX Titanium) platforms.

Materials and Methods

EDITTAG provides a suite of Python (http://www.python.org) programs for: validating sequence tags for conformance to the edit or Hamming distance metrics, designing edit metric sequence tags, and incorporating sequence tags to amplicons or platform-specific sequencing adapters. We describe implementation details for each of these EDITTAG processes, and we follow each description with the steps we followed to implement or validate each process.

Sequence Tag Validation

The validate_edit_metric_tags.py program within EDITTAG checks existing tag sets, alone or incorporated into PCR primers or sequencing adapters, for conformance to the edit metric by performing pairwise, edit distance comparisons between each tag in the input set and all other tags in the set. In short, the program iterates through the set of tags input; computes the pairwise edit distance between all tags in the set using either a C-based Python module or a pure-Python method; and outputs either the minimum distance of the set, those tag pairs having an edit distance less than the minimum expected, or the edit distance between all members of a set, depending on the output options selected by the user. This program is also capable of computing the Hamming distance between sequence tag inputs based on selection of the Hamming algorithm in place of the edit distance algorithm by the user.

We used validate_edit_metric_tags.py to test the conformance of eight existing sequence tag sets available from commercial (Illumina, Inc. and Roche 454, Inc.) and non-commercial sources [29,31,40–42] to their respective distance metric (Hamming or edit) by appropriately formatting an input file for these tags (File S1) and inputting this file to the program. We used the tag-rescanning feature of design_edit_metric_tags.py (described below) to determine the number of tags in these sequence tags sets having minimum edit distances of three and five.

Sequence Tag Design

Technically, designing error-correcting sequence tags is a matter of generating all *n*-length combinations of [A,C,G,T]; filtering tags based on subjective or platform-specific criteria including removal of: combinations containing homopolymer runs, combinations with undesirable base composition, or individual tags that are perfect self-complements; and iteratively comparing each tag in the remaining group against all other tags in the remaining group to create the largest set that maintains some minimum edit distance. Practically, the process is more complex because the design of sequence tag sets requires comparison of all tags in the candidate set to all other tags in the candidate set. Given sequence tags of sufficient length, this requirement rapidly approaches the limits of desktop computation.

Table 1. Commercial and non-commercial sequence tag sets and the conformance of each to the stated or assumed distance metric (edit or Hamming).

Class	Set Name	Length (nt)	N_{tags}	Design Algorithm	Minimum Distance		Pair Violations	Tags≥$D_{expected}$	Comments
					exp	obs			
Contain Violations	Illumina TruSeq sRNA	6	48	Hamming	3	2	2	47	Some tags violate expected Hamming distance
	Hamady et al. 2007[1]	8	1544	Hamming	3	4/2[2]	-	1544	Only corrects 66% of errors
	Meyer et al. 2010[3]	6	75	Edit	3	2	40	49	Some tags violate expected edit distance
	Meyer et al. 2010[3]	8	711	Edit	3	2	551	429	Some tags violate expected edit distance
	Adey et al. 2010[4]	9	96	Edit	4	2	58	64	Some tags violate expected edit distance
Correct Hamming distance	Illumina TruSeq RNA and DNA	6	27	Hamming	3	3	-	27	
	Meyer et al. 2008[5]	7	52	Hamming	3	3	-	52	
	Meyer et al. 2008[5]	8	130	Hamming	3	3	-	130	
Correct edit distance	Qiu et al. 2003	6	21	Edit	3	3	-	21	
	Frank 2009[6]	6	81	Other	2	2	-	81	Design algorithm similar to edit distance 2
	Illumina Nextera DNA[7]	8	8/12	Edit	3	3	-	8 or 12	
	Frank 2009[2]	8	760	Other	2	2	-	760	Design algorithm similar to edit distance 2
	Roche 454 MID Extended	10	151	Edit	4	4	-	151	
	Roche 454 RL-MID Extended	10	132	Edit	4	4	-	132	
Designed for this publication	EDDITTAG	6	61	Edit	3	3	-	61	
	EDDITTAG	7	211	Edit	3	3	-	211	
	EDDITTAG	8	531	Edit	3	3	-	531	
	EDDITTAG	9	1,936	Edit	3	3	-	1,936	
	EDDITTAG	10	7,198	Edit	3	3	-	7,198	

[1]Hamady *et al.* [31] tags are from the nmeth.1184-S1.pdf supplementary file.
[2]Hamady *et al.* [31] tags are Hamming distance 4 from one another in binary encoding but Hamming distance 2 from one another in nucleotide encoding.
[3]We generated Meyer *et al.* [29] tags using: 'python create_index_sequences.py -l <length> -d 3'.
[4]Adey *et al.* [41] tags are from the gb-2010-11-12-r119-s3.pdf supplementary file.
[5]Meyer *et al.* [3] tags are from the nprot.2007.520-S1.doc supplementary file.
[6]We generated Frank [42] tags using: 'barcrawl -l <length> -m 3'. BARCRAWL uses a hybrid approach to create distance between tags while accounting for a single deletion. This is similar to an expected edit distance of two.
[7]Illumina Nextera tags are incorporated to either end of the template strand in combinatorial fashion to identify up to 96 samples.
doi:10.1371/journal.pone.0042543.t001

For example, the full set of 10 nucleotide tags contains 1,048,576 members, which requires 550 billion pairwise edit distance comparisons across all tags in the candidate set. If storage of each result requires 8 bits, then storing the entire array requires approximately 500 GB - a daunting object with which to work. Additionally, this considers only the first stage of processing and ignores the additional computational and storage overhead required to select and test subsets of edit metric sequence tags.

Thus, we modified the approach used by the lexicode algorithm [43] to speed up processing, reduce memory consumption, and enable parallelization of jobs across multiple processors. Briefly, our approach first generates all *n*-length combinations of [A,C,G,T]. Then, if the remaining group is sufficiently large, we apportion tags into discrete batches of 25,000 tags, and we distribute each batch among the available number of processing cores to (optionally) remove those tags having problematic composition (homopolymers, improper GC, perfect self-complements). After filtering, we rebuild the set of candidate tags returned from each processing core, and we create the following data structure, where the 0th position of each "row" below is a sequence tag "key" to which we pair a "value" comprising a list of all tags in the set:

```
(
  (tag0,[(tag0),(tag1),(tag2),(tag3)]),
  (tag1,[(tag0),(tag1),(tag2),(tag3)]),
  ...
)
```

If this data structure is sufficiently long (more than 500 "rows" as illustrated above), we apportion the structure into batches containing 500 "rows", and we distribute each batch among the available number of processors. Iterating over each row, we then compute the edit distance between the "key" and all sequence tags in the value list using either a C-based Python module (http://pylevenshtein.googlecode.com) or a pure-Python method. To reduce memory consumption when iterating over millions of tags, we produce a summary vector for each key giving the count of all other sequence tags having values that fall within edit distance categories (0, 1, 2, …, *N*), and we use the 0-indexed position of the count in the vector to denote the edit distance. Thus, the vector:

([1,12,124,5])

corresponds to a key having a single tag edit distance 0 from the key, 12 tags edit distance one from the key, 124 tags edit distance two from the key, and five tags edit distance three from the key. We then reduce the data by keeping only those keys having the maximum count of comparisons at the minimum desired edit distance, a technique that allows us to reduce the remaining number of pairwise comparisons over the entire data set by approximately 99% (estimated from the generation of eight nucleotide, edit distance three tags).

After reducing the data, for each key we compute the edit distance between the key and all sequence tags in the value; we drop any tags in the value less than the desired edit distance; and we iterate over the remaining tags in the value, retaining only those tags that are also the desired edit distance from one another. Finally, we determine the count of remaining tags in the value list for each key, and we return the key (and its values) having the largest value list. Additionally, we include an option that quickly returns subsets of keys within this final set having edit distances from the key at values greater than the minimum desired edit distance.

We used this approach to design sets of edit metric sequence tags ranging from four to 10 nucleotides in length and having edit distances of three. We used the shortcut method described above to select subsets, within each of these sets, having edit distances from four to nine. After creating these edit distance tags, we validated each set of resulting tags for conformance to the edit metric using validate_edit_metric_tags.py, the program described in the previous subsection.

Sequence Tag Application

EDITTAG provides two convenience programs for integrating sequence tags to platform-specific adapters and PCR primers. The first program (add_tags_to_primers.py) is meant primarily for integration of sequence tags to PCR amplicons when designing sequence-tagged PCR primers. In brief, this program adds sequence tags to the 5′ ends of both upper and lower PCR primers, optionally removes common bases between each sequence tag and primer sequence, optionally prepends both primers with a sequence (GTTT) promoting +A addition [44] to facilitate adapter ligation, uses Primer3 [45] to evaluate tagged primers for complementarity problems and the presence of hairpins, and outputs all tagged primers to an sqlite (http://www.sqlite.org) database or comma-separated file for subsequent evaluation and selection.

The second program (add_tags_to_adapters.py) simply integrates designed sequence tags to adapters and/or primers by inputting the list of desired sequence tags, the adapter/primer sequence 5′ of the sequence tag location, and the adapter/primer sequence 3′ of the sequence tag location. This program is largely meant to reduce mistakes when manually positioning sequence tags within large numbers of adapters or primers.

Testing Sequence Tag Integration to PCR Primers

To test the design and resulting utility of PCR primers sequence-tagged using the helper program, we integrated the entire set ($n = 164$) of 10 nucleotide, edit distance five sequence tags (File S2) to primers amplifying the *rbcLa* locus in land plants [46,47]. We used the resulting database to select 95 hairpin-free, sequence tagged primers (File S3) which we had commercially synthesized, adding a single 3′ phosphorothioate linkage to each oligo (Integrated DNA Technologies, Inc.). We used these primers to amplify the *rbcLa* locus in 190 tropical forest tree species (2×95 reactions) in a reaction mixture containing 5.0 µL CTAB-extracted [48], purified (AMPure) DNA, 0.3 µM KAPA dNTP mix, 0.2 µM each primer, 1× KAPA HiFi PCR Buffer, 0.5 U KAPA HiFi HotStart polymerase and the following touchdown PCR thermal profile: 95°C for 30 s; 20 cycles of 95°C for 30 s, 66°C for 30 s minus 0.25°C per cycle, 72°C for 1.5 min; 20 cycles of 95°C for 30 s, 60°C for 30 s, 72°C for 1.5 m; 72°C for 15 min. Following PCR, we visualized amplicons by running 7 µL of PCR product on 1.5% agarose gels for 90 minutes at 100 V and staining with ethidium bromide.

We cleaned PCR amplicons and normalized amplicon concentrations across samples using SequalPrep normalization plates (Invitrogen, Inc.), combined sequence-tagged PCR amplicons from a 96-well plate into a single pool, and concentrated the pool using a SpeedVac. Prior to sequencing, we used T/A ligation to add standard 454 GS FLX Titanium sequencing adapters to the 5′ and 3′ ends of each amplicon pool [49]. We quantified the resulting adapter-ligated amplicon pools using qPCR (KAPA Biosystems), we combined amplicon pools at equimolar ratios, and

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.